AIDA Debugging

Using Emacs JDEE

Starting an aida server in JDEE (JDE->Debug App)

M-x customize-variable

- jde-db-option-vm-args. Note the DEV in these - change it to PROD for a production server.
  `-Xbootclasspath/p:/afs/slac/package/iona/orbacus/prod/JOB/lib/OB.jar`
  `-DAIDA_DATABASE_USERID=AIDADEV`
  `-DAIDA_NAMESERVER_IOR_FILE=/afs/slac.stanford.edu/package/aida/slaconly/NameServerDEV.ior`
  `-DAIDA_NAMESERVER_IOR_URL=http://www.slac.stanford.edu/grp/cd/soft/slaconly/aida/NameServerDEV.ior`

Orbacus Diagnostics

Both Aida servers, and the test clients (test.java) pass on command line arguments to the ORB init. So one can use, for instance `-ORBtrace_connections` to diagnose Orbacus: Eg Using the Aida command line aliases to start a server, specify level 3 tracing with:

```
aidaproc ${AIDA_NAME_NAME} ${AIDA_NAME_DBGADDR} sys.daNameServer.NameServer -ORBtrace_connections 3
```

Test clients from test.java also pass on -ORB options, but any arguments are interpreted as to the test client, the following will execute tests 1 and 3, not use level 3 tracing:

```
aidatest daDpTestHistTest 1 -ORBtrace_connections 3
```

VMS Data Provider Debugging

Java Part of VMS Data Providers

Usual jdk works, so edit the startdp<server>.com file, replacing the java command at the end with jdk, or just do the setup yourself and run the server all from teh command line:

```
MCCDEV> url = "-DAIDA_NAMESERVER_IOR_URL=http://www.slac.stanford.edu/grp/cd/sof
t/slaconly/aida/NameServer''f$trnlnm("AIDA_MODE")'.ior"
MCCDEV> conf == "-Dooc.config=DPSLCBPM.CONF"
MCCDEV> bootcp = "-Xbootclasspath/p:/DISK$SLCLIBRARY/JAVA/FOREIGN/OB.JAR"
MCCDEV> jdb $bootcp $uid $url $conf "edu.stanford.slac.aida.dp.dpSlcBpm.DpSlcBpm
Server
Initializing jdb ...>
```

Native code part of VMS Data Providers

The regular VMS debugger works fine for debugging the native code parts of an Aida server running in an interactive process (which for a production aida server is not the norm). So, start the server in a decterm, and then do the usual ctrl-y debug on it to set break points. The following example shows this for debugging the main data acquisition function (DPSLCBPM_BPMACQ) of the DPSLCBPM data provider:

```
DPSLCBpm general_init went OK.
Tue Aug 15 09:52:01 GMT-08:00 2006: Server ready
 Interrupt <-- CTRL-Y

MCC> debug
%DEBUG-I-SSINOTSET, system service interception is not setup, defaulting to nost
atic watchpoints
%DEBUG-I-NOGLOBALS, some or all global symbols not accessible

         OpenVMS Alpha Debug64 Version V7.3-200


DBG> set image aidashr
DBG>
LI06  BPM FDBK MBCD stat = 00000000, expected = 49930000.  CR= 4, MO=19.
DBG> set modu dpslcbpm_jni_helper
DBG> set sco DPSLCBPM
LI06  BPM FDBK MBCD stat = FFA30002, expected = 39930000.  CR= 3, MO=19.
DBG> set sco DPSLCBPM_BPMACQ
DBG> set break DPSLCBPM_BPMACQ
DBG> go
```

When debugging the dpslc*_jni.c module the names of the routines are very long, since they're constructed by JNI and include the whole aida package name. In this case use the debugger's "sho sym *routineprefix*" to search for the symbol corresponding to the routine's name. The name may well have been truncated, so start with the first few letters of the name that you'd expect. Once you have the name you can use the usual set break <name>.

Now, when a client (possibly from unix) asks for some data from the server, the VMS code will break at the specified location in your VMS terminal session.